Implementation and Automatic Testing for Security Enhancement of Linux Based on Least Privilege

Gaoshou Zhai, Jie Zeng, Miaoxia Ma, Liang Zhang
School of Computer and Information Technology, Beijing Jiaotong University
gszhai@bjtu.edu.cn

Abstract

Nowadays, technologies of information security have been attached more and more importance to and it's a critical problem to take measures to ensure the reliability of related trustworthy software such as secure operating systems (SOSs). Thereafter, it's always necessary for such systems to be taken complete and rigorous security test and evaluation among development team and/or by third-party security certification organization. However, such software testing is usually time consuming, cost consuming and boresome and thus technologies of software testing automation have alluring application foreground in that field. In this paper, methods and technologies about how to test a SOS automatically are discussed in breadth and in depth at first. Then least privilege is studied and the corresponding modules of security enhancement are added to Linux based on Linux Kernel Modules (LKM). Finally, a prototype of automatic security testing as to such least privilege mechanism is implemented and the results are analyzed.

1. Introduction

It's well-known that information security is supported and safeguarded by security of information systems whose infrastructure are made up of operating systems and hardware architectures. Thus security of operating systems are being attached more and more importance to and great effort are taken to build up secure operating systems (SOSs) or consolidate security of current operating systems by introducing series of secure mechanisms. Meanwhile, it's a critical problem to take measures to ensure the reliability of secure operating systems and it's necessary for them to be taken complete and rigorous security test and evaluation among development team and/or by third-party security certification organization. However, such software testing is usually time consuming, cost consuming and boresome and thus technologies of software testing automation have alluring application foreground in that field[1-3]. In the following sections of this paper, methods and technologies about how to test a SOS automatically are discussed at first, including extensions of security testing with varied automatic testing methods, framework of whole automatic security testing platform for SOS and modular design for automatic security testing of security-related system calls. Then least privilege is studied and the corresponding modules of security enhancement are added to Linux based on LKM. Finally, a prototype of automatic security testing as to such least privilege mechanism is implemented and the results are analyzed.

2. Automatization of security testing for secure operating systems

Security test is a special kind of software test and it is focused on those security-supported functions in secure operating systems.

2.1. Extensions of security testing with varied automatic testing methods

From the view of modern software engineering, software test is an activity through the entire software developing procedure. Thereafter, security test ought to start from requirement analysis stage and span to final usage stage, including requirement evaluation and validation, design evaluation and validation, security check of code, unit testing, integrated testing, acceptance testing, system testing and etc (refer to figure 1). At the same time, test objects encompass not only source codes and execution system, but also documents, procedure management of development and etc.

Meanwhile, methods of formal validation, informal validation and simulated intrusion detection can be used to implement security testing, and simulated intrusion detection is generally a compensatory method to be used to take an ulterior step to search for more security faults. Furthermore, formal validation methods can be used to test a SOS built up based on formal requirement specification and formal design methods, or just to test the security model of a SOS; while informal validation methods, e.g. reviewing, discussion, validation and system testing, can be used to test a SOS built up based on informal methods. Obviously, system testing is essential for all SOSs.

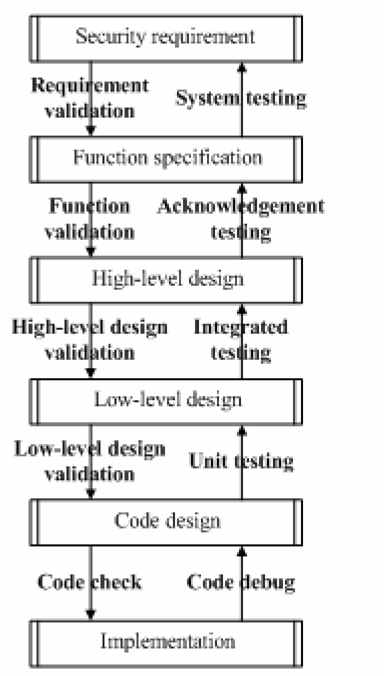

Figure 1. **Extensions of security testing for SOS**

At the different stages of security testing, various automatic measures can be adopted. For example, integrated software engineering management tools can be used to implement automatic check of coherence among different development documents and to improve reviewing quality; formal verification tools can be used to implement automatic verification of formal security models; security leaks scanning tools or intruder expert can be used to accelerate the process of finding secure hidden trouble inside the system. In addition, automatic security testing platform can be built up and be used to implement systematic security test of SOS, as will be further discussed in the following sections of this paper.

2.2. Framework of whole automatic security testing platform for SOS

Automatic security testing platform for SOS can be made up of documents consistence check module, system-call integrated tester, intrusion detection module, command testers, system call testers, configuration checker, security audit tester and security function configuration interface (refer to Figure 2).

Command testers and system call testers are to test security related commands and system calls respectively and both of them can apply data driven methods. While system-call integrated tester is attempt to test combination of system calls in different sequences. All of these three testers need to build up precondition environment and provide expected system security states. Obviously, system-call integrated testers are difficult but critical to be built up.

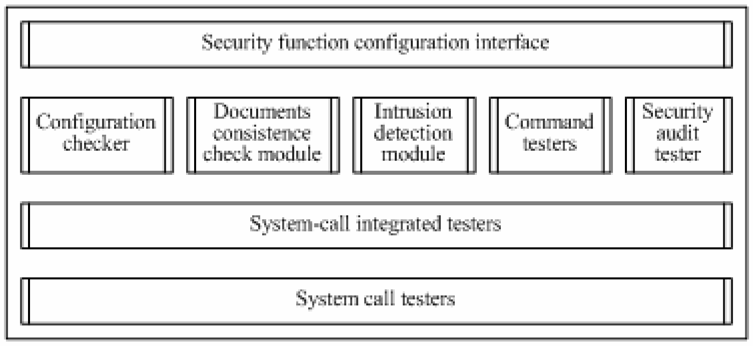
Security function configuration interface
Configuration checker
Documents consistence check module
Intrusion detection module
Command testers
Security audit tester
System-call integrated testers
System call testers

Figure 2. **Framework of automatic security testing platform for SOS**

Documents consistence check module aims at check consistence among different specification and documents and formal security model verifier is included. And it depends on integrated computer aided software engineering tools.

Intrusion detection module is design to detect vulnerability of the system and intruder simulation modules are important. Security audit tester is to test security audit function of the system. Configuration checker is to check maturity, sufficiency and consistence among different configuration items. While security functions configuration interface can be connected to security evaluation/certificate platforms and integrated developing environment so as to be reused farthest.

It is well-known that security-supported functions of a SOS are mainly executed and in effect through its user interface, including interactive user interface (e.g. command-like interface, graphic user interface) and program interface (i.e. system call interface), which shows that command testers and system call testers are the most important components of the whole automatic security testing platform. At the same time, construction rationale and mechanism for these two parts are similar. Therefore, system call testers will be focused in this paper.

2.3. Modular design for system call testers

The basic steps to test a system call and modular structure of a system call tester can be referred to figure 3 and figure 4 respectively.

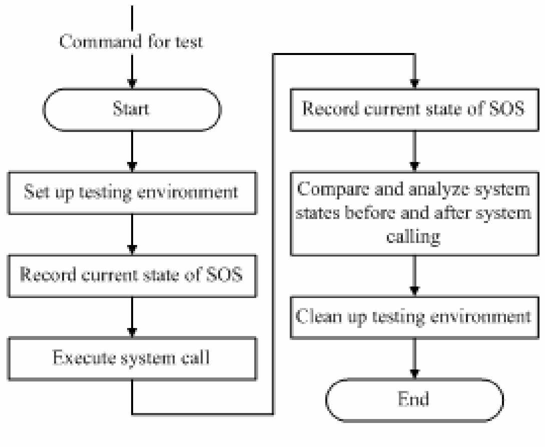
Command for test
Start
Set up testing environment
Record current state of SOS
Execute system call
Record current state of SOS
Compare and analyze system states before and after system calling
Clean up testing environment
End

Figure 3. **Basic steps to test a system call**

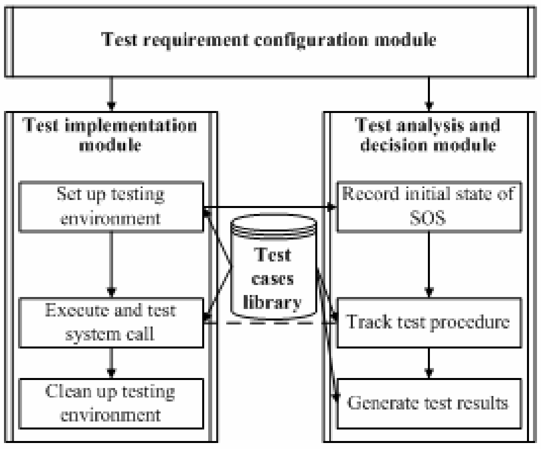
Test requirement configuration module
Test implementation module
Set up testing environment
Execute and test system call
Clean up testing environment
Test cases library
Test analysis and decision module
Record initial state of SOS
Track test procedure
Generate test results

Figure 4. **Modular structure of a system call tester**

3. Security enhancement as to Linux based on least privilege

In order to carry out automatic security test and to take a further study, a prototype of SOS is built up based on Linux by introducing least privilege mechanism in the way of Linux Kernel Modules (LKM).

3.1. Least privilege mechanism

Moreover, the prototype adopts RBAC model[4] and least privilege is implemented based on privilege states of process, where an improved algorithm is put forward and adopted for privilege computing and transferring by the way of learning from other's strong points to offset one's weakness among a series of available algorithms such as POSIX[5], Linux 2.2.0[6], LinSec[7], DG/UX[8] and etc. The algorithm can be described as follows:

(1) pI* = pI | pIadd
(2) pE' = (fP | (fI & pI*))) & fE & pB & gB
(3) pI' = pI* & fI & fB
(4) pB' = pB & fB

where pI stands for inheritable privilege set of current process within which privileges can be transferred to its successive process images; pIadd stands for a special structure that contains information of life cycle and age for each added privilege which can be used to control transferring generations of the privilege; fP stands for permitted privilege set of program file within which privileges are necessary for correct execution of function as to the corresponding process; fI stands for inheritable privilege set of program file within which privileges can be transferred from its corresponding process to the successive process images under some other conditions; fE stands for effective privilege set of program file that can be used to decide effective privilege set of its corresponding process; pB and fB stand for binding privilege set of a process or program file respectively which contains maximal privileges that a process or program file can contain at most; gB stands for global binding privilege set that is made up of all privileges that can be owned by any process in the system; pE', pI', pB' have similar signification as pE, pI and pB with the only difference that privilege sets are owned by the corresponding successive process images.

In addition, some system calls and other interfaces are inserted into the system for convenient switch of process privilege states and they can be called by process according to its needs. Meanwhile, privileges of original super manager are divided and allocated to some roles that are restricted one another, and cross authorization as to some security sensitive operation is implemented by the way of digital signature.

3.2. Architecture of the SOS prototype

The architecture of the SOS prototype can be referred to figure 5, where SEM, SAA, SCL stand for security enhanced modules, system assistant applications (e.g. secret key generation tools, signature tools for ELF files and tasks) and security configuration library respectively.

3.3. Implementation of the SOS prototype

The prototype is implemented in C language for both user level and kernel level programs, and LKM technologies are adopted for kernel level programming.

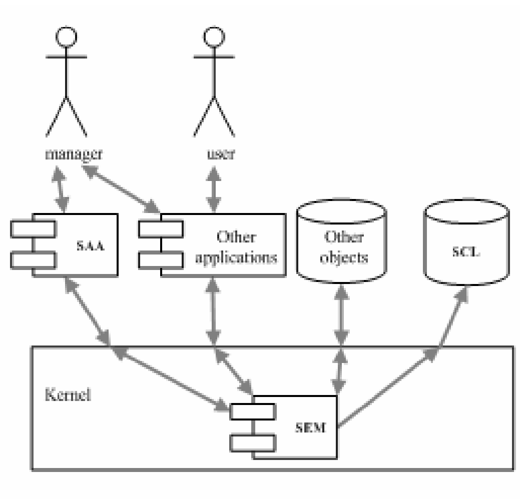

Figure 5. **Architecture of the SOS prototype**

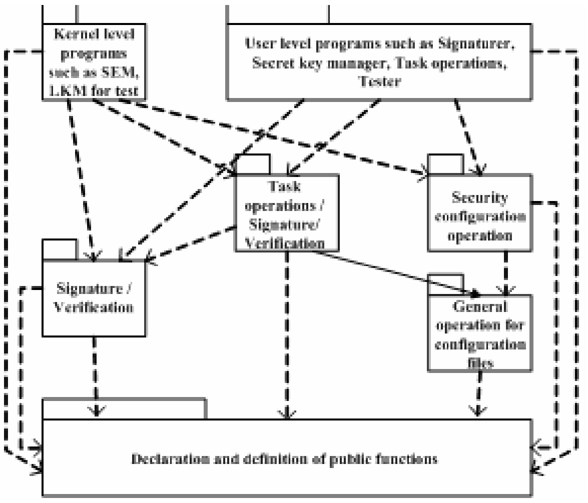
Kernel level programs such as SEM, LKM for test
User level programs such as Signaturer, Secret key manager, Task operations, Tester
Task operations / Signature/ Verification
Security configuration operation
Signature / Verification
General operation for configuration files
Declaration and definition of public functions

Figure 6. **Architecture of the SOS prototype**

The modular package structure of the SOS prototype can be referred to figure 6, where reuse of modules and functions are fully considered and incarnated.

In addition, all of general configuration files such as signature files, task files, security configuration files and etc adopt a file format similar to Windows INI.

4. Automatic security testing of least privilege mechanisms

The key for automatic security testing is to analyze the testing target and to build up the automatic testing platform.

4.1. Analysis of testing target

The testing target (i.e. target of evaluation or TOE) is least privilege mechanisms based on LKM and it make comprehensive use of RBAC, privilege transfer between privilege states of process and cross authorization. Considering that it is implemented by intercepting, capturing and modifying some system calls such as mkdir, open, close, execve, create_module and delete_module, security testing ought to be focused on those system calls and check if functions of these and other related system calls can be played correctly so as to verify if the availability of the entire system is weakened under the conditions of satisfying security strategies. Moreover, it ought to be verified if privileges of each process satisfy security strategies during the procedure of system evolution. Correctness of cross authorization for each role on program files and correctness of task signature and execution should be also verified.

Four different roles of security manager, system operator, auditor and security operator are set up and six users are configured for the system. Furthermore, four users are act as each role respectively, one user is ordinary user and one user owns four roles at the same time.

4.2. Design and implementation of automatic security testing platform

The automatic security testing platform for least privilege mechanisms can be made up of execution program for test scripts (i.e. runscript), loadable kernel module for test (i.e. tkn.o), test scripts (or scripts for test), assistant files for test and scripts for test

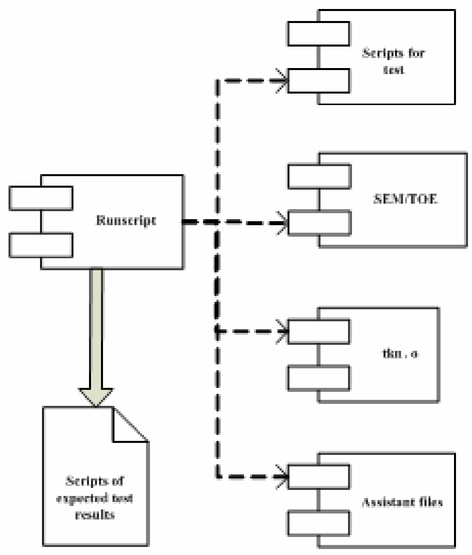

Figure 7. **Components of automatic security testing platform**

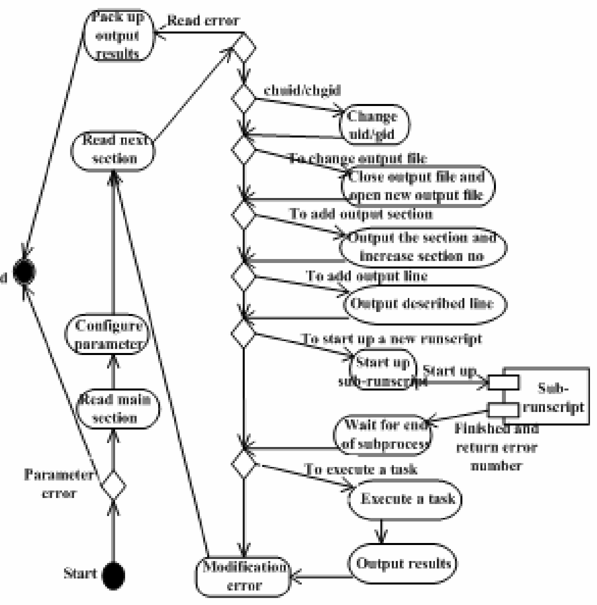

Figure 8. **Architecture of the SOS prototype**

results (refer to figure 7). Note that SEM/TOE is referred to the test target i.e. least privilege mechanisms.

Runscript is responsible for reading test scripts and assistant files for test, executing corresponding operations and making use of security enhanced modules including tkn.o, and generating scripts for test results finally (refer to figure 8). And it ought to have several copies with different security configurations so as to test privilege transferring features.

Several copies are also required for tkn.o. One copy is signed by security manager and it should be able to be installed and another copy is signed by other manager than security manager and it should not be able to be installed. Also a copy without any signature is needed and it should not be able to be installed obviously. Furthermore, another copy signed by both security manager and other managers is needed and it should be able to be installed. Note that both TOE and tkn.o have intercepted and captured the system call of mkdir, the order of load and unload can't be intercrossed. In other words, first installed ought to be uninstalled at last, last installed ought to be uninstalled at first.

4.3. Test and results

Test procedure starts from executing runscript in the status of root and it can be divided into following steps:

(1) Set up testing environment, e.g. install SEM/TOE, configure roles and users, create assistant files for test and etc.

(2) Execute testing on process privilege and process execution before install SEM.

(3) Execute testing on installing SEM

(4) Execute testing on role privilege, process privilege, process privilege transfer, cross-authorization and execution, cross-authorization and task, and SEM uninstalling etc after install SEM.

Test results show that no error occur for testing before installing SEM and installing itself. It is also verified that after SEM is installed, expected errors are returned under the condition of exceeding one's authority for access testing as to roles, process privilege transfer and cross-authorization and for execution testing as to task of cross-authorization. And testing for SEM uninstalling proves that any role in the status of root can't uninstall SEM individually and only security manager and system operator execute uninstall operations in order can SEM be uninstalled successfully.

5. Summary

Automatization of security testing for SOS is studied. A SOS is constructed based on Linux according to least privilege principle and RBAC model by LKM technologies and a prototype of automatic security testing for it is built up. Then, the security enhanced modules are tested by execute that tester prototype and the test procedure and results are satisfactory. It shows that automatic security testing methods for SOS put forward in this paper are feasible and effective. And they can be improved and extended to be used to build up automatic security testing platform for entire SOS and implement deep security tests in the next step. In addition, Automatization of generating test cases based on security requirement and/or security evaluation criteria and/or protected profiles can be further studied in future.

6. Acknowledgements

The research presented in this paper was performed with the support of Beijing Pivot Discipline Construction Project for Computer Application Technologies at Beijing Jiaotong University for XK100040519 and Scholastic Fund of Beijing Jiaotong University for 2005SM016.

7. References

[1] [1] Gaoshou Zhai, Zeng Jie, Miaoxia Ma, Liang Zhang, “Automatization of security testing as to secure operating system”, Proceedings of CNCC 2007, Tsinghua university printing house, 2007.(in Chinese)

[2] Pihui Wei, Sihan Qing, Jian Huang, "An evaluation system for secure operating system", Computer Engineering, vol.29, no.22, 2003, pp.135-137.(in Chinese)

[3] Youli Lu, Hongqi Zhang, "Design of system for security testing and evaluation as to operating systems", Information security and communication secrecy, no.8, 2005, pp.94-97.(in Chinese)

[4] R.S.Sandhu, et al. “Role Based Access Control Models”, IEEE Computer 29(2): 38-47, IEEE Press, 1996.

[5] Portable Applications Standards Committee of IEEE Computer Society. Standards Project, Draft Standard for Information Technology-Portable Operating System Interface (POSIX). PSSG Draft 17, New York: IEEE, Inc, 1997.

[6] Charistias P.J. Unix Online Man Pages: Capability(4). Http://www.mcsr.olemiss.edu/cgi-bin/man-cgi?Capabilities, 1994

[7] Knight G. LinSec-Linux security protection system. Http://www.linsec.org/doc/final/final.pdf. 2002.

[8] General D. “Managing Security on DG/UX System”. Tech Rep: 093-701138-09, Westboro, MA: Data General, A Division of EMC Corporation, 2001.

Authors

Gaoshou Zhai, Ph.D. Now he is working as an associate professor at school of Computer and Information Technology, Beijing Jiaotong University, Beijing, China. His research interests include operating systems, information security, system software design and automatic tools for software engineering.